

A Mini Project Report
On
**Prediction of Used Car Prices Using Artificial Neural Networks
And Machine Learning**

Submitted to JNTU HYDERABAD

In Partial Fulfillment of the requirements for the Award of Degree of

**BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE AND ENGINEERING**

Submitted

By

M.Durgaprasad	208R1A05F8
Md.Abdul Adil	208R1A05G0
E.Tulasi	208R1A05D8
P.Niharika	208R1A05G4

Under the Esteemed guidance of

Mrs S.Anitha

Associate Professor, Department of CSE



Department of Computer Science & Engineering

CMR ENGINEERING COLLEGE

(Approved by AICTE, NEW DELHI, Affiliated to JNTU, Hyderabad)
Kandlakoya, Medchal Road, R.R. Dist. Hyderabad-501 401)

2023-2024

CMR ENGINEERING COLLEGE

(Accredited by NBA, Approved by AICTE NEW DELHI, Affiliated to JNTU, Hyderabad)

Kandlakoya, Medchal Road, Hyderabad-501 401

Department of Computer Science & Engineering



CERTIFICATE

This is to certify that the project entitled “**Prediction of Used Car Prices Using Artificial Neural Networks And Machine Learning**” is a bonafide work carried out by

M.Durgaprasad	208R1A05F8
Md.Abdul Adil	208R1A05G0
E.Tulasi	208R1A05D8
P.Niharika	208R1A05G4

in partial fulfillment of the requirement for the award of the degree of **BACHELOR OF TECHNOLOGY** in **COMPUTER SCIENCE AND ENGINEERING** from CMR Engineering College, affiliated to JNTU, Hyderabad, under our guidance and supervision.

The results presented in this project have been verified and are found to be satisfactory. The results embodied in this project have not been submitted to any other university for the award of any other degree or diploma.

Internal Guide

Mrs S.Anitha
Associate Professor
CSE, CMREC

Mini Project Coordinator

Mr. S. Kiran Kumar
Assistant Professor
CSE, CMREC

Head of the Department

Dr. Sheo Kumar
Professor & H.O.D
CSE, CMREC

DECLARATION

This is to certify that the work reported in the present project entitled " **Prediction of UsedCar Prices Using Artificial Neural Networks And Machine Learning**" is a record of bonafide work done by us in the Department of Computer Science and Engineering, CMR Engineering College, JNTU Hyderabad. The reports are based on the project work done entirely by us and not copied from any other source. We submit our project for further development by any interested students who share similar interests to improve the project in the future.

The results embodied in this project report have not been submitted to any other University or Institute for the award of any degree or diploma to the best of our knowledge and belief.

M.Durgaprasad	208R1A05F8
Md.Abdul Adil	208R1A05G0
E.Tulasi	208R1A05D8
P.Niharika	208R1A05G4

ACKNOWLEDGMENT

We are extremely grateful to **Dr. A. Srinivasula Reddy**, Principal and **Dr. Sheo Kumar**, HOD, **Department of CSE, CMR Engineering College** for their constant support.

We are extremely thankful to **Mrs S Anitha**, Associate Professor, Internal Guide, Department of CSE, for his/ her constant guidance, encouragement and moral support throughout the project.

We will be failing in duty if I do not acknowledge with grateful thanks to the authors of the references and other literatures referred in this Project.

We thank **S Kiran Kumar** Mini Project Coordinator for his constant support in carrying out the project activities and reviews.

We express my thanks to all staff members and friends for all the help and co-ordination extended in bringing out this project successfully in time.

Finally, we are very much thankful to my parents who guided me for every step.

M.Durgaprasad	208R1A05F8
Md.Abdul Adil	208R1A05G0
E.Tulasi	208R1A05D8
P.Niharika	208R1A05G4

TOPIC	PAGE-NO
ABSTRACT	7
1. INTRODUCTION	8
2. LITERATURE SURVEY	9-10
3. SYSTEM ANALYSIS	
3.1 Existing System & Disadvantages	11
3.2 Proposed System & Advantages	12
3.3 Functional Requirements	13
3.4 Non-Functional Requirements	14
4. SYSTEM REQUIREMENTS	15
4.1 Hardware Requirements	
4.2 Software Requirements	
5. SOFTWARE DESIGN	
5.1 System Overview	16-17
5.2 System Architecture	18
5.3 Unified Modelling Language Diagrams	19
5.3.1 Use Case Diagram	20
5.3.2 Activity Diagram	21
5.3.3 Sequence Diagram	22
5.3.4 Class Diagram	
6. SYSTEM IMPLEMENTATION	23-35
6.1 Evaluation Methodology	23 -24
6.1.1 Modifying Dataset	

TOPIC	PAGE -NO
6.2 Modules	25
6.3 Machine Learning	
6.3.1 Support Vector Machine Algorithm	26
6.3.2 Random forest Algorithm	26
6.3.3 K-nearest Neighbours	26
6.4 Technologies	27-35
6.4.1 Python	
6.4.2 Flask Web Framework	
7. CODING	36-39
8. OUTPUT SCREENS	40-44
9. TESTING	45-47
9.1 System Testing	48
10. CONCLUSION	49
11. Future Enhancement	50-51
12. REFERENCE	52

ABSTRACT

It is generally known that, taking wise and challenging decisions is really a crucial task in every business. Taking improper decisions can cause huge loss and even lead to shutdown of business. To propose a novel solution for this challenge, this research work majorly focuses on one of the retail businesses i.e., used car sales business. The proposed research work shows that, the predictive analytical models will be a great add-on to business mainly for assisting the decision making process. Predictive Analytics is a process, where the businesses use statistical methods and technologies to analyze their historical data for delivering new insights and plan the future accordingly. The major objective of our paper is to build a prediction model i.e., a fair price mechanism to predict the cars selling price based on their features like the car model, the number of years that a car is old, the type of fuel it uses, the type of seller, the type of transmission and the number of kilometers that the car has driven so far. This paper will help to get an approximation about selling price of a used car based on its features and reduces the seller and consumer risk in business. The proposed model utilizes the machine learning algorithms and regression techniques of statistics like linear, decision tree and random forest regressions to achieve this task.

1.INTRODUCTION

In recent years, the market for used cars has witnessed a significant boom due to various economic factors, changing consumer preferences, and advancements in technology. As a result, accurately predicting the prices of used cars has become a crucial task for buyers, sellers, and dealers alike. Traditional methods of pricing used cars often fall short in providing precise and reliable estimates due to the complex interplay of numerous factors. With the advent of Artificial Neural Networks (ANNs) and Machine Learning (ML) techniques, the field of predicting used car prices has seen remarkable progress. ANNs, inspired by the human brain's neural networks, and ML algorithms offer the ability to analyze vast datasets, identify intricate patterns, and make predictions with a high degree of accuracy. This interdisciplinary approach combining computer science, statistics, and automotive expertise has revolutionized the way used car prices are determined.

The primary objective of this research is to develop a robust and efficient model for predicting used car prices using Artificial Neural Networks and Machine Learning algorithms. By harnessing the power of deep learning and advanced ML techniques, this study aims to create a predictive framework that can analyze various features of used cars and accurately estimate their market value.

The methodology employed in this research involves collecting a comprehensive dataset comprising information about used cars, including make, model, year of manufacture, mileage, fuel type, transmission type, and other relevant features. This dataset forms the basis for training the Artificial Neural Network and Machine Learning models. The integration of Artificial Neural Networks and Machine Learning techniques in predicting used car prices represents a significant advancement in the automotive industry. This research endeavors to leverage these technologies to enhance accuracy and efficiency, thereby shaping a more reliable and transparent used car market.

2.Literature survey

1. Prediction of Used Car Prices Using Machine Learning Techniques" by M. I. A. Miah, M. S. Arefin, and M. S. Islam. This paper presents a comparative study of different machine learning algorithms, including artificial neural networks, for predicting used car prices. The study uses a dataset of more than 100,000 used car listings and compares the performance of different models in terms of accuracy, speed, and computational efficiency.(2022)
2. Predicting Used Car Prices Using Neural Networks" by A. F. Rahman, M. A. Uddin, and M. M. Islam. This paper presents a neural network-based approach for predicting used car prices. The study uses a dataset of more than 10,000 used car listings and evaluates the performance of the model in terms of accuracy and computational efficiency.(2020)
3. Gegic, E. et al. (2019) demonstrate the need to create a model to forecast the cost of second hand cars in Bosnia and Herzegovina. They used machine learning techniques such as artificial neural networks, support vector machines, and random forests. However, the aforementioned methods were used in concert. The web scraper, which was created using the PHP programming language, was used to gather the data from the website autopijaca.ba for the forecast. Then, to determine which method best suited the provided data, the respective performances of various algorithms were compared. A Java application contained the final prediction model. Additionally, the model's accuracy of 87.38% was determined when it was verified using test data. Dholiya et al. demonstrated a machine learning-based method for auto resales in 2019.
4. Predicting the value of a used car is an intriguing topic that has been studied extensively. Sameerch and Pudaruth employed decision trees, K-nearest neighbors, linear regression, and Naive Bayes to predict used automobile prices from newspaper data. This model takes into account only a few automotive models and is 61% correct. Furthermore, Asghar et al used a statistical test to select the best qualities and a linear regression to estimate the price of a secondhand car. Noor and Jan used multiple linear regression to create a model for predicting car prices. They considered a number of elements, including price, capacity,

color, ad date, power steering, kilometers, and so on. Following feature selection, the authors examined only the engine type, price, model year, and car model as input features. In his thesis work, Richardson also employed multiple regression analysis and discovered that hybrid cars retain their value for a longer period of time than normal automobiles. This arises from environmental concerns and it also improves fuel efficiency. The work presented in Gonggi takes the following factors into account mileage, manufacturer, and expected useful life. The model was fine-tuned to account for nonlinear relationships.

5. Listiani published another study that is comparable and uses Support Vector Machines (SVM) to forecast lease car pricing . This study demonstrated that when a very large data set is available, SVM is significantly more accurate at price prediction than multiple linear regression. SVM is also superior at handling high dimensional data and steers clear of both under- and over-fitting problems. Finding crucial features for SVM is done using a genetic algorithm. However, the method does not demonstrate why SVM is superior to basic multiple regression in terms of variance and mean standard deviation.
6. The aim of the study is to use Linear Regression (LR) algorithm based price prediction of car price and accuracy comparison with support vector machine (SVM) classification algorithm. Materials and methods: LR (N=205) and SVM algorithm (N=205) are applied for car price prediction as a mechanism. The accuracy and prediction of the classifiers was evaluated and recorded with G power 80% and alpha value 0.05. Results: The SVM produces 89% accuracy in predicting the car price on the sample dataset and the LR predicts the accuracy at the rate 91.7%. LR algorithm based accuracy appears (significant 0.563) to be better than SVM algorithm for car price prediction. Conclusion: The accuracy performance parameter of the LR algorithm appears to be better than the SVM algorithm.(2022)

3.System Analysis

3.1 EXISTING SYSTEM:

Considering the demand for private car all around the world, the demand of second-hand car market has been rising and creating a chance in business for both buyer and seller. In several countries, buying a used car is the best choice for customer because its price is reasonable and affordable by buyer. After few years of using them, it may get a profit from resell again. However, various factors influence the price of a used car such as how old of those vehicles and the condition in current scenario of them. Normally, the price of used cars in the market is not constant. Thus, car price evaluation model is required for helping in trading.

Predicting the price of second-hand cars has not received much attention from academia despite its huge importance for the society. Bharambe and Dharmadhikari (2015) used artificial neural networks (ANN) to analyse the stock market and predict market behaviour. They claimed that their proposed approach is more accurate than existing ones by 25%. Pudaruth (2014) used four different supervised machine learning techniques namely kNN (kNearest Neighbour), Naïve Bayes, linear regression and decision trees to predict the price of second-hand cars. The best result was obtained using kNN which had a mean error of 27000 rupees. Jassbi et al. (2011) used two different neural networks and regression methods to predict the thickness of paint coatings on cars. The error for the final thickness of the paint was found to be 2/99 microns for neural networks and 17/86 for regression. Ahangar et al. (2010) also compared the use of neural networks with linear regression in order to predict the stock prices of companies in Iran. They also found that neural networks had superior performance both in terms of accuracy and speed compared to linear regression. Listiani (2009) used support vector machines (SVM) to predict the price of leased cars.

DISADVANTAGES:

- ❖ **Limited Data Availability:** The accuracy of the predictions generated by the model heavily relies on the quality and quantity of data available for training. If the available data is incomplete or not representative of the market, the predictions may not be accurate.
- ❖ **Model Complexity:** Artificial neural networks can be very complex, and their behavior is not always easy to interpret. This can make it challenging to understand the factors that are

driving the model's predictions, which can make it difficult to improve or refine the model over time.

- ❖ **Model Overfitting:** Overfitting occurs when a model is trained too well on the training data and becomes less effective at predicting new, unseen data. This can occur if the model is too complex or if there is too little data available for training.
- ❖ **Maintenance and Updates:** Artificial neural networks and machine learning models require regular maintenance and updates to ensure that they remain accurate and relevant. This can be time-consuming and expensive, and it may be difficult to justify the investment for smaller organizations or businesses.

3.2 PROPOSED SYSTEM:

The proposed research work shows that, the predictive analytical models will be a great add-on to business mainly for assisting the decision making process. Predictive Analytics is a process, where the businesses use statistical methods and technologies to analyze their historical data for delivering new insights and plan the future accordingly. The major objective of our paper is to build a prediction model i.e., a fair price mechanism to predict the cars selling price based on their features like the car model, the number of years that a car is old, the type of fuel it uses, the type of seller, the type of transmission and the number of kilometers that the car has driven so far. This paper will help to get an approximation about selling price of a used car based on its features and reduces the seller and consumer risk in business. The proposed model utilizes the machine learning algorithms and regression techniques of statistics like linear, decision tree and random forest regressions to achieve this task.

In order to carry out this study, data have been obtained from different car websites and from the small found in daily newspapers like L'Express and Le Defi. • The data was collected in less than one month interval because like other goods, the price of cars also changes with time. • Two hundred records were collected. The data comprises of different features for second-hand cars such as the year in which it was manufactured, the make, engine capacity measured in cubic centimetres, paint type, transmission type, mileage and its price in Mauritian rupees. • A large number of experiments have been conducted in order to find the best network structure and the best parameters for the neural network. We found that a neural network with 1 hidden layer and

2 nodes produced the smallest mean absolute error among various neural network structures that were experimented with. • However, we found that Support Vector Regression and a multilayer perception with back-propagation produced slightly better predictions than linear regression while the k-Nearest Neighbour algorithm had the worst accuracy among these four approaches. All experiments were performed with a cross validation value of 10 folds

ADVANTAGES:

- ❖ Accuracy: Artificial neural networks and machine learning algorithms are capable of accurately predicting used car prices. These techniques can analyze large amounts of data and identify patterns that may not be apparent to human analysts, resulting in more accurate price predictions.
- ❖ Efficiency: Once trained, artificial neural networks and machine learning algorithms can make predictions quickly and efficiently. This allows car dealerships and other businesses to make pricing decisions in real-time, improving their ability to compete in the market.
- ❖ Scalability: Machine learning algorithms can be applied to large datasets, allowing car dealerships and other businesses to analyze a wide range of factors that affect used car prices, such as make and model, year, mileage, and condition.
- ❖ Flexibility: Artificial neural networks and machine learning algorithms can be trained to predict prices for different types of used cars and can adapt to changes in the market, such as fluctuations in demand or changes in the economy.

3.3 Functional Requirements

1. Data Collection and Integration

- The system collects a diverse dataset of used car listings from various sources.
- Data from different sources are integrated into a unified format for analysis.

2. Data Preprocessing

- The system handles missing values, outliers, and inconsistencies in the data
- Data normalization and feature scaling are applied uniformly across the dataset.

3. Model Selection and Training

- The system offers a variety of machine learning algorithms and neural network architectures.
- It automates the training process and optimizes models using techniques such as hyperparameter tuning and cross-validation.

4. Evaluation and Validation:

- The system evaluates models using appropriate metrics like MAE, MSE, or RMSE.
- Visualizations and reports are provided to help users understand the model's performance.

5. Prediction and Interpretability:

- Users can input new car details and receive price predictions.
- The system provides explanations or feature importance scores to interpret predictions, enhancing user understanding.

3.4 NON-FUNCTIONAL REQUIREMENTS

1. Performance

- Real-time or near-real-time predictions
- Ability to handle large volumes of simultaneous users and data points efficiently

2. Reliability:

- Consistently accurate predictions
- Graceful error handling and recovery mechanisms

3. Scalability

- Capability to handle a growing dataset and increasing user load
- Horizontal scalability by adding more computational resources or nodes

4. Security

- Robust security measures to protect user data and prevent unauthorized access
- User authentication and authorization mechanisms for data privacy

5. Usability:

- Intuitive and user-friendly interface for input and interpretation of predictions
- Clear and understandable explanations for predicted prices

6. Maintainability

- Modularity and clean code practices for ease of maintenance and updates
- Automation of maintenance tasks such as model retraining and data updates

7. Compatibility

- Compatibility with various platforms, browsers, and devices
- Seamless integration with different data sources and APIs

8. Compliance

- Compliance with relevant laws, regulations, and industry standards
- Adherence to ethical guidelines regarding user data and machine learning algorithms

9. Documentation

- Comprehensive documentation for system administrators, developers, and end-users
- Includes installation guides, user manuals, and API documentation

10. Performance Monitoring and Logging

- Logging and monitoring capabilities for user interactions, system usage, and performance metrics
- Regular performance audits and system health checks

4.SYSTEM REQUIREMENTS

4.1 HARDWARE REQUIREMENTS:

- **Processor** - Intel Core i5 or AMD Ryzen 5 (or higher)
- **RAM** - 8 GB (min)
- **Hard Disk** -20 GB
- **Key Board** -Standard Windows Keyboard
- **Mouse** -Two or Three Button Mouse
- **Monitor** -SVGA

SOFTWARE REQUIREMENTS:

- **Operating system** : Windows 7,8,10,11.
- **Coding Language** : Python
- **Front-End** : Python.
- **Back-End** : Django-ORM.
- **Designing** : HTML, CSS, JAVASCRIPT.
- **Data Base** : MySQL (WAMP Server).

5.SOFTWARE DESIGN

5.1System Overview

Data Gathering: The source of the data is the web portal of Kaggle.com where vehicle dataset of carde kho is provided for selling and buying of cars. The dataset gave the following set of features: Car Name, Year, Selling Price, Present or the Current Price, Kilometers driven, Fuel Type: Petrol, Diesel or CNG (Compressed Natural Gas), Seller Type: Dealer or Individual, Transmission: Automatic or Manual, Owner (No. of previous owners).

	name	company	year	Price	kms_driven	fuel_type
0	Hyundai Santro Xing XO eRLX Euro III	Hyundai	2007	80,000	45,000 kms	Petrol
1	Mahindra Jeep CL550 MDI	Mahindra	2006	4,25,000	40 kms	Diesel
2	Maruti Suzuki Alto 800 Vxi	Maruti	2018	Ask For Price	22,000 kms	Petrol
3	Hyundai Grand i10 Magna 1.2 Kappa VTVT	Hyundai	2014	3,25,000	28,000 kms	Petrol
4	Ford EcoSport Titanium 1.5L TDCi	Ford	2014	5,75,000	36,000 kms	Diesel

Creating Environment: An environment is created using anaconda prompt. This environment would separate our project space from the other default(base) or any other environments created previously. All the packages, libraries and modules that we require can be manually installed in the environment created using this manner and this makes this a beneficial step. We can make the changes according to our requirements in such an environment.


```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 816 entries, 0 to 815
Data columns (total 6 columns):
#   Column      Non-Null Count  Dtype
---  -
0   name        816 non-null   object
1   company     816 non-null   object
2   year        816 non-null   int32
3   Price       816 non-null   int32
4   kms_driven  816 non-null   int32
5   fuel_type   816 non-null   object
dtypes: int32(3), object(3)
memory usage: 28.8+ KB

```

```

car.describe(include='all')
✓ 0.0s

```

	name	company	year	Price	kms_driven	fuel_type
count	816	816	816.000000	8.160000e+02	816.000000	816
unique	254	25	NaN	NaN	NaN	3
top	Maruti Suzuki Swift	Maruti	NaN	NaN	NaN	Petrol
freq	51	221	NaN	NaN	NaN	428
mean	NaN	NaN	2012.444853	4.117176e+05	46275.531863	NaN

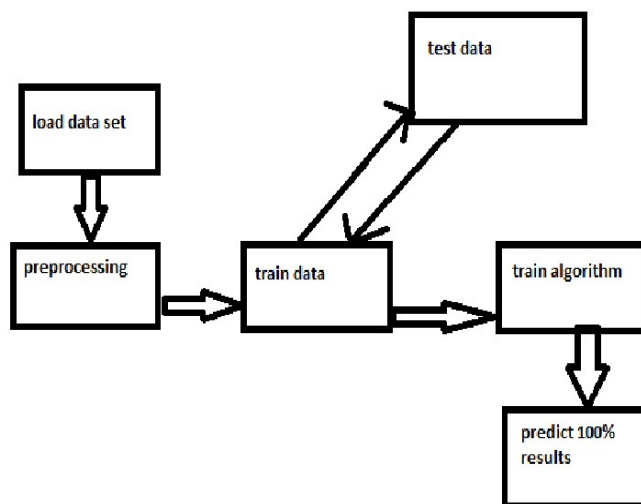
Data Reading: The csv file is imported and read for the study which is the primary step. The dataset is thoroughly read on various aspects like null values, shape, columns, numerical and categorical features, dataset columns, unique values of each feature, data info etc.

Data Pre-processing: Some of the features in the data were renamed (Present Price = Initial Price, Owner = Previous Owners) for better understanding and some other features that were not useful for analysis were also dropped. Exploratory Data Analysis of data is done in which we use statistical graphics and other visualization methods to summarize the main characteristics of data. Various graphs and charts such as: Top Selling vehicles, Year v/s vehicles available, Selling Price v/s Initial Price, Vehicle Fuel Type, Transmission Type, Seller Type, Age, Selling Price v/s Age, Selling Price v/s Seller Type, Selling Price v/s Transmission, Selling Price v/s Fuel Type, Selling Price v/s Previous Owners, Initial Price vs Selling Price, Selling Price v/s Kilometers Driven, pairplot, heatmaps etc. are plotted to get a better understanding of data. After completing EDA, One Hot Encoding technique is employed for dealing with the categorical features of the dataset. Thereafter, the correlation features of the dataset are produced and analyzed thoroughly by visualizing some plots. Then the features allocation of data is done where the dependent feature(Selling Price) and independent features(Initial Price, Kilometers Driven, Previous Owners, Age etc.) are allocated for further procedure.

Train-Test Split: After the allocation of dependent and independent features is completed, we proceed further with the splitting of dataset into training and testing data. We use 80% of data for training our model and 20% data for testing purposes.

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 816 entries, 0 to 815  
Data columns (total 6 columns):  
#   Column      Non-Null Count  Dtype  
---  ---  
0   name        816 non-null    object  
1   company     816 non-null    object  
2   year        816 non-null    int32  
3   Price       816 non-null    int32  
4   kms_driven  816 non-null    int32  
5   fuel_type   816 non-null    object  
dtypes: int32(3), object(3)  
memory usage: 28.8+ KB
```

5.2 SYSTEM ARCHITECTURE:

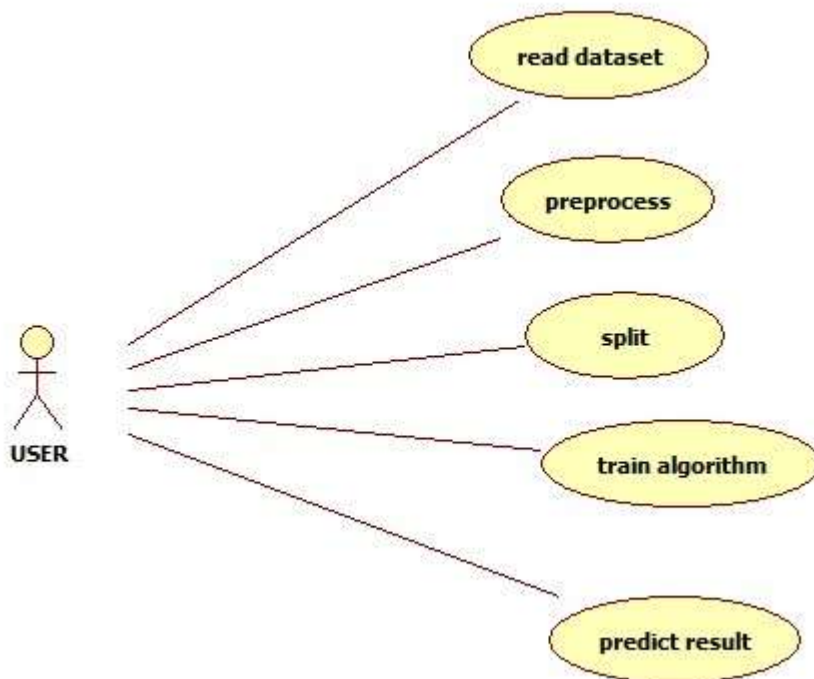


5.3 UNIFIED MODELLING LANGUAGE DIAGRAMS

UML is a method for describing the system architecture in detail using the blue print. UML represents a collection of best engineering practice that has proven successful in the modeling of large and complex systems. The UML is very important parts of developing object-oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects. Using the helps UML helps project teams communicate explore potential designs and validate the architectural design of the software.

5.3.1 USE CASE DIAGRAM

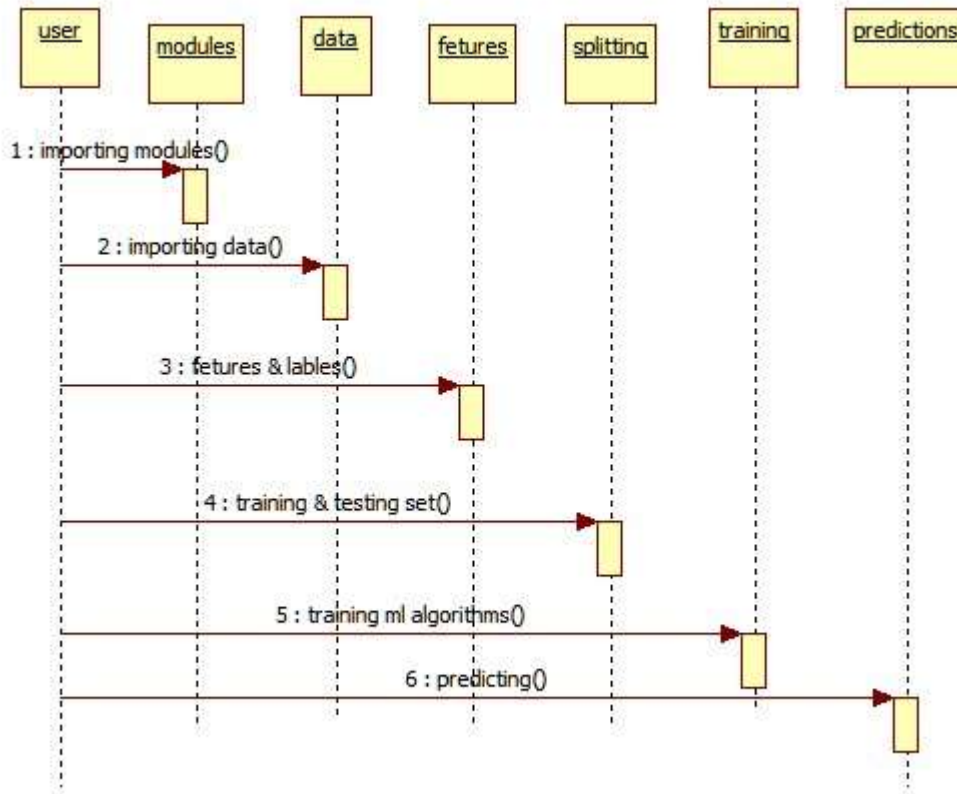
A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted



USE CASE DIAGRAM

5.3.2 ACTIVITY DIAGRAM

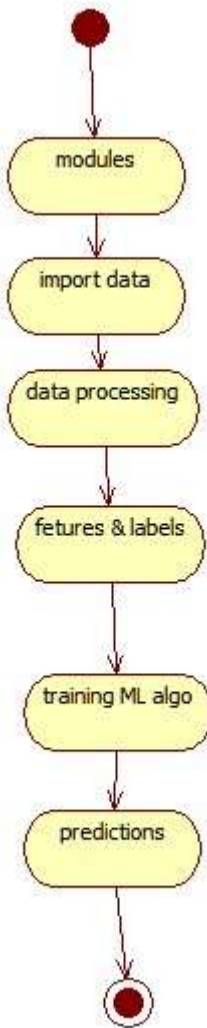
Activity diagrams are graphical representations of work flows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step work flows of components in a system. An activity diagram shows the overall flow of control.



ACTIVITY DIAGRAM

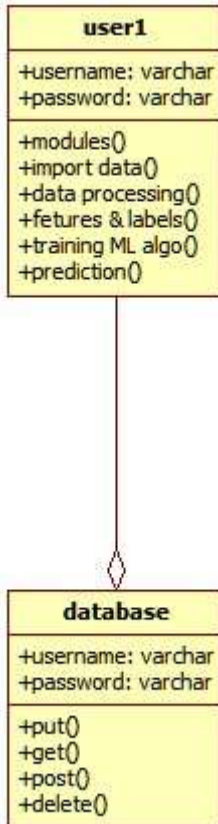
5.3.3 SEQUENCE DIAGRAM

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagram.



5.3.4 CLASS DIAGRAM

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.



6SYSTEM IMPLEMENTATION

6.1Evaluation Methodology

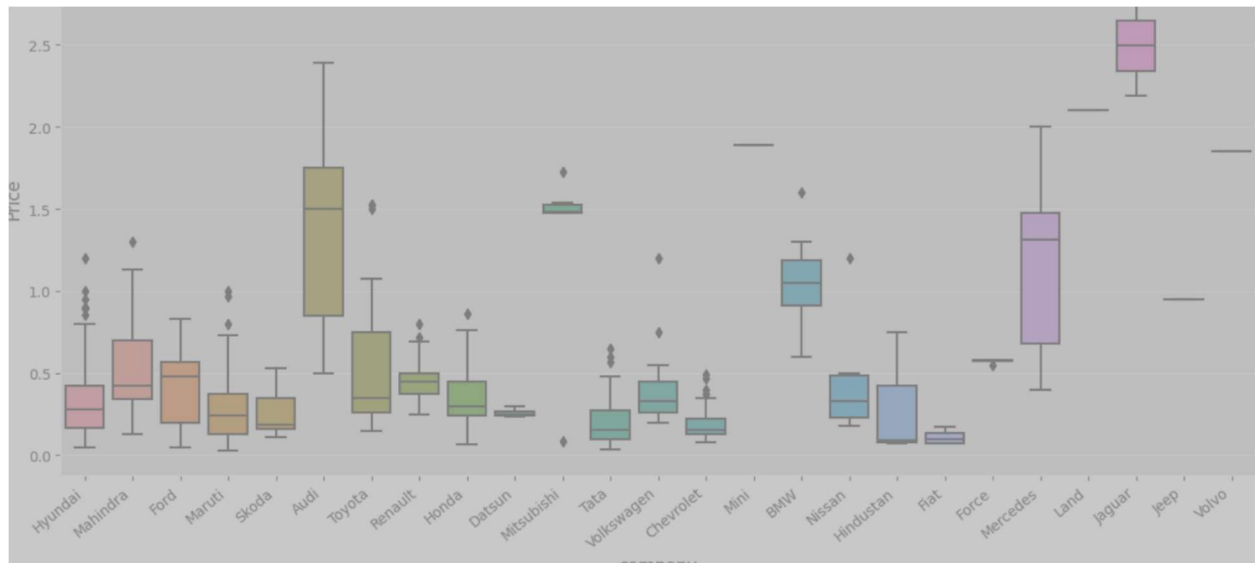
6.1.1Modifying Dataset

Creating a new feature Age which determines the number of years the vehicle has been used for and storing it into final dataset and removing the year attribute.

	name	company	year	Price	kms_driven	fuel_type
count	816	816	816.000000	8.160000e+02	816.000000	816
unique	254	25	NaN	NaN	NaN	3
top	Maruti Suzuki Swift	Maruti	NaN	NaN	NaN	Petrol
freq	51	221	NaN	NaN	NaN	428
mean	NaN	NaN	2012.444853	4.117176e+05	46275.531863	NaN
std	NaN	NaN	4.002992	4.751844e+05	34297.428044	NaN
min	NaN	NaN	1995.000000	3.000000e+04	0.000000	NaN
25%	NaN	NaN	2010.000000	1.750000e+05	27000.000000	NaN
50%	NaN	NaN	2013.000000	2.999990e+05	41000.000000	NaN
75%	NaN	NaN	2015.000000	4.912500e+05	56818.500000	NaN
max	NaN	NaN	2019.000000	8.500003e+06	400000.000000	NaN

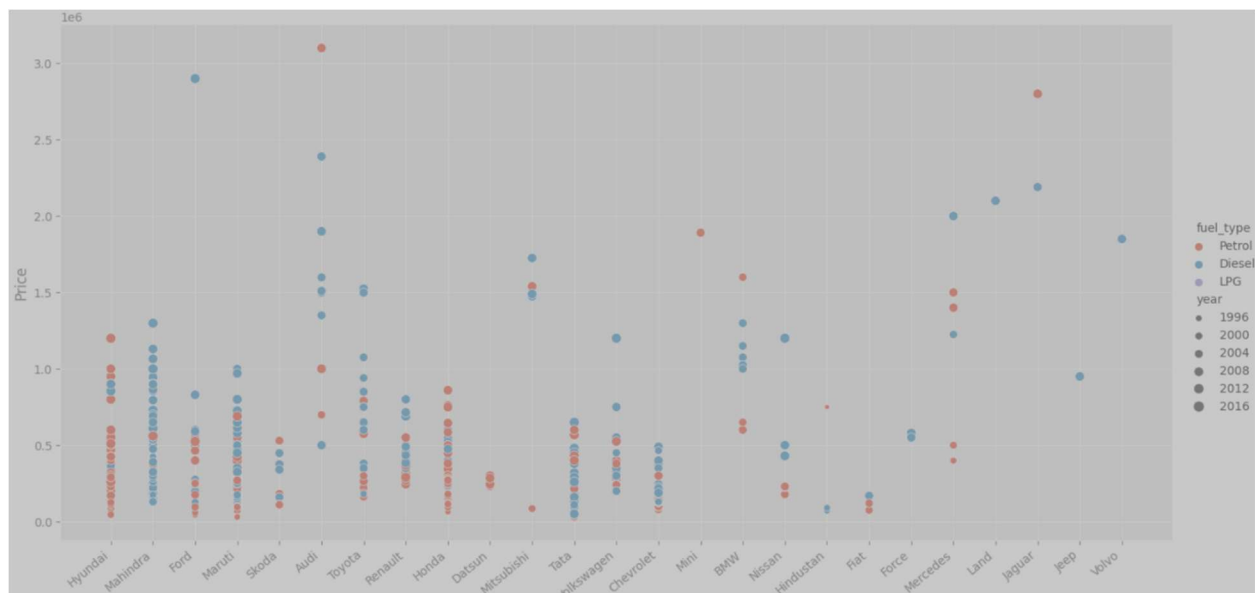
Exploratory Data Analysis of data is done in which we use statistical graphics and other visualization methods to summarize the main characteristics of data. Various graphs and charts are plotted to get a better understanding of the dataset as well as the relationship of features in dataset.

Checking relationship of Company with Price



Checking relationship of Year with Price

When checking the relationship between the year of a car and its price, you're essentially exploring how the price of a used car changes based on its manufacturing year. Here are the general steps you can take to analyze this relationship



Applying Train Test Split

Applying the train-test split is a crucial step in machine learning to assess how well your model generalizes to new, unseen data. Here's how you can apply the train-test split to your dataset for building a linear regression model to predict used car prices based on the manufacturing year

```
Pipeline(memory=None,
          steps=[('columntransformer',
                  ColumnTransformer(n_jobs=None, remainder='passthrough',
                                    sparse_threshold=0.3,
                                    transformer_weights=None,
                                    transformers=[('onehotencoder',
                                                  OneHotEncoder(categories=[array(['Audi A3 Cabriolet', 'Audi A4 1.8', 'Audi A4 2.0', 'Audi A6 2.0',
'Audi A8', 'Audi Q3 2.0', 'Audi Q5 2.0', 'Audi Q7', 'BMW 3 Series',
'BMW 5 Series', 'BMW 7 Series', 'B...
'Renault', 'Skoda', 'Tata', 'Toyota', 'Volkswagen', 'Volvo'],
dtype=object),
                                                  array(['Diesel', 'LPG', 'Petrol'], dtype=object)],
                                                  drop=None,
                                                  dtype=<class 'numpy.float64'>,
                                                  handle_unknown='error',
                                                  sparse=True),
                                                  ['name', 'company',
'fuel_type'])),
                  verbose=False)),
          ('linearregression',
           LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
                             normalize=False))),
          verbose=False)
```

6.2 MODULES:

Load Dataset:

Load data set using pandas read_csv() method.

Split Data Set:

Split the data set to two types. One is train data test and another one is test data set.

Train data set:

Train data set will train our data set using fit method.

Test data set:

Test data set will test the data set using algorithm.

Predict data set:

Predict() method will predict the results.

6.3.1 SUPPORT VECTOR MACHINE ALGORITHM:

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning. The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane. SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine. Consider the below diagram in which there are two different categories that are classified using a decision boundary or hyperplane.

6.3.2 RANDOM FOREST ALGORITHM:

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model. As the name suggests, "Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output. The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.

6.3.3 K-Nearest Neighbors (KNN) ALGORITHM:

K-Nearest Neighbors (KNN) is a supervised machine learning algorithm used for classification and regression tasks. It works based on the principle that similar data points are close to each other in a feature space. In the context of predicting used car prices, you can use KNN for regression to estimate the price of a car based on its features and the features of its nearest neighbors.

6.4 TECHNOLOGIES

PYTHON

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. An [interpreted language](#), Python has a design philosophy that emphasizes code [readability](#) (notably using [whitespace](#) indentation to delimit [code blocks](#) rather than curly brackets or keywords), and a syntax that allows programmers to express concepts in fewer [lines of code](#) than might be used in languages such as [C++](#) or [Java](#). It provides constructs that enable clear programming on both small and large scales. Python interpreters are available for many [operating systems](#). [CPython](#), the [reference implementation](#) of Python, is [open source](#) software and has a community-based development model, as do nearly all of its variant implementations. CPython is managed by the non-profit [Python Software Foundation](#). Python features a [dynamic type](#) system and automatic [memory management](#). It supports multiple [programming paradigms](#), including [object-oriented](#), [imperative](#), [functional](#) and [procedural](#), and has a large and comprehensive [standard library](#).

Interactive Mode Programming

Invoking the interpreter without passing a script file as a parameter brings up the following prompt

—

```
$ python
```

```
Python 2.4.3 (#1, Nov 11 2010, 13:34:43)
```

```
[GCC 4.1.2 20080704 (Red Hat 4.1.2-48)] on linux2
```

```
Type "help", "copyright", "credits" or "license" for more information.
```

```
>>>Type the following text at the Python prompt and press the Enter —
```

```
>>> print "Hello, Python!"
```

If you are running new version of Python, then you would need to use print statement with parenthesis as in `print ("Hello, Python!")`; However in Python version 2.4.3, this produces the following result —Hello, Python!

Script Mode Programming

Invoking the interpreter with a script parameter begins execution of the script and continues until the script is finished. When the script is finished, the interpreter is no longer active.

Let us write a simple Python program in a script. Python files have extension .py. Type the following source code in a test.py file –

Live Demo

```
print "Hello, Python!"
```

We assume that you have Python interpreter set in PATH variable. Now, try to run this program as follows –

```
$ python test.py
```

This produces the following result –

```
Hello, Python!
```

Let us try another way to execute a Python script. Here is the modified test.py file –

Live Demo

```
#!/usr/bin/python
```

```
print "Hello, Python!"
```

We assume that you have Python interpreter available in /usr/bin directory. Now, try to run this program as follows –

```
$ chmod +x test.py # This is to make file executable
```

```
$/test.py
```

This produces the following result –

```
Hello, Python!
```

Python Identifiers

A Python identifier is a name used to identify a variable, function, class, module or other object. An identifier starts with a letter A to Z or a to z or an underscore (_) followed by zero or more letters, underscores and digits (0 to 9).

Python does not allow punctuation characters such as @, \$, and % within identifiers. Python is a case sensitive programming language. Thus, Manpower and manpower are two different identifiers in Python.

Here are naming conventions for Python identifiers –

Class names start with an uppercase letter. All other identifiers start with a lowercase letter.

Starting an identifier with a single leading underscore indicates that the identifier is private.

Starting an identifier with two leading underscores indicates a strongly private identifier.

If the identifier also ends with two trailing underscores, the identifier is a language-defined special name.

Reserved Words

The following list shows the Python keywords. These are reserved words and you cannot use them as constant or variable or any other identifier names. All the Python keywords contain lowercase letters only.

```
and    exec    not
assert finally or
break  for     pass
class  from    print
continue      global raise
def     if     return
del     import try
elif    in     while
else    is     with
except lambda yield
```

Lines and Indentation

Python provides no braces to indicate blocks of code for class and function definitions or flow control. Blocks of code are denoted by line indentation, which is rigidly enforced.

The number of spaces in the indentation is variable, but all statements within the block must be indented the same amount. For example –

```
if True:
    print "True"
else:
    print "False"
```

However, the following block generates an error –

```
if True:
print "Answer"
print "True"
else:
print "Answer"
print "False"
```

Thus, in Python all the continuous lines indented with same number of spaces would form a block. The following example has various statement blocks –

Note – Do not try to understand the logic at this point of time. Just make sure you understood various blocks even if they are without braces.

```
#!/usr/bin/python
```

```
import sys
```

```
try:
    # open file stream
    file = open(file_name, "w")
except IOError:
    print "There was an error writing to", file_name
    sys.exit()
```

```

print "Enter ", file_finish,
print " When finished"
while file_text != file_finish:
    file_text = raw_input("Enter text: ")
    if file_text == file_finish:
        # close the file
        file.close
        break
    file.write(file_text)
    file.write("\n")
file.close()
file_name = raw_input("Enter filename: ")
if len(file_name) == 0:
    print "Next time please enter something"
    sys.exit()
try:
    file = open(file_name, "r")
except IOError:
    print "There was an error reading file"
    sys.exit()
file_text = file.read()
file.close()
print file_text

```

Multi-Line Statements

Statements in Python typically end with a new line. Python does, however, allow the use of the line continuation character (\) to denote that the line should continue. For example –

```

total = item_one + \
        item_two + \
        item_three

```

Statements contained within the [], {}, or () brackets do not need to use the line continuation character. For example –

```

days = ['Monday', 'Tuesday', 'Wednesday',
        'Thursday', 'Friday']

```

Quotation in Python

Python accepts single ('), double (") and triple (" or """) quotes to denote string literals, as long as the same type of quote starts and ends the string.

The triple quotes are used to span the string across multiple lines. For example, all the following are legal –

```

word = 'word'
sentence = "This is a sentence."
paragraph = """This is a paragraph. It is
made up of multiple lines and sentences."""

```

Comments in Python

A hash sign (#) that is not inside a string literal begins a comment. All characters after the # and up to the end of the physical line are part of the comment and the Python interpreter ignores them.

Live Demo

```
#!/usr/bin/python
```

```
# First comment
```

```
print "Hello, Python!" # second comment
```

This produces the following result –

```
Hello, Python!
```

You can type a comment on the same line after a statement or expression –

```
name = "Madisetti" # This is again comment
```

You can comment multiple lines as follows –

```
# This is a comment.
```

```
# This is a comment, too.
```

```
# This is a comment, too.
```

```
# I said that already.
```

Following triple-quoted string is also ignored by Python interpreter and can be used as a multiline comments:

```
'''
```

```
This is a multiline  
comment.
```

```
'''
```

Using Blank Lines

A line containing only whitespace, possibly with a comment, is known as a blank line and Python totally ignores it.

In an interactive interpreter session, you must enter an empty physical line to terminate a multiline statement.

Waiting for the User

The following line of the program displays the prompt, the statement saying “Press the enter key to exit”, and waits for the user to take action –

```
#!/usr/bin/python
```

```
raw_input("\n\nPress the enter key to exit.")
```

Here, "\n\n" is used to create two new lines before displaying the actual line. Once the user presses the key, the program ends. This is a nice trick to keep a console window open until the user is done with an application.

Multiple Statements on a Single Line

The semicolon (;) allows multiple statements on the single line given that neither statement starts a new code block. Here is a sample snip using the semicolon.

```
import sys; x = 'foo'; sys.stdout.write(x + '\n')
```

Multiple Statement Groups as Suites

A group of individual statements, which make a single code block are called suites in Python. Compound or complex statements, such as if, while, def, and class require a header line and a suite. Header lines begin the statement (with the keyword) and terminate with a colon (:) and are followed by one or more lines which make up the suite. For example –

```
if expression :  
    suite  
elif expression :  
    suite  
else :  
    suite
```

Command Line Arguments

Many programs can be run to provide you with some basic information about how they should be run. Python enables you to do this with -h –

```
$ python -h  
usage: python [option] ... [-c cmd | -m mod | file | -] [arg] ...  
Options and arguments (and corresponding environment variables):  
-c cmd : program passed in as string (terminates option list)  
-d      : debug output from parser (also PYTHONDEBUG=x)  
-E      : ignore environment variables (such as PYTHONPATH)  
-h      : print this help message and exit
```

You can also program your script in such a way that it should accept various options. Command Line Arguments is an advanced topic and should be studied a bit later once you have gone through rest of the Python concepts.

Python Lists

The list is a most versatile datatype available in Python which can be written as a list of comma-separated values (items) between square brackets. Important thing about a list is that items in a list need not be of the same type.

Creating a list is as simple as putting different comma-separated values between square brackets. For example –

```
list1 = ['physics', 'chemistry', 1997, 2000];  
list2 = [1, 2, 3, 4, 5];  
list3 = ["a", "b", "c", "d"]
```

Similar to string indices, list indices start at 0, and lists can be sliced, concatenated and so on.

A tuple is a sequence of immutable Python objects. Tuples are sequences, just like lists. The differences between tuples and lists are, the tuples cannot be changed unlike lists and tuples use parentheses, whereas lists use square brackets.

Creating a tuple is as simple as putting different comma-separated values. Optionally you can put these comma-separated values between parentheses also. For example –

```
tup1 = ('physics', 'chemistry', 1997, 2000);  
tup2 = (1, 2, 3, 4, 5 );  
tup3 = "a", "b", "c", "d";
```

The empty tuple is written as two parentheses containing nothing –

```
tup1 = ();
```

To write a tuple containing a single value you have to include a comma, even though there is only one value –

```
tup1 = (50,);
```

Like string indices, tuple indices start at 0, and they can be sliced, concatenated, and so on.

Accessing Values in Tuples

To access values in tuple, use the square brackets for slicing along with the index or indices to obtain value available at that index. For example –

Live Demo

```
#!/usr/bin/python
```

```
tup1 = ('physics', 'chemistry', 1997, 2000);  
tup2 = (1, 2, 3, 4, 5, 6, 7 );  
print "tup1[0]: ", tup1[0];  
print "tup2[1:5]: ", tup2[1:5];
```

When the above code is executed, it produces the following result –

```
tup1[0]: physics  
tup2[1:5]: [2, 3, 4, 5]
```

Updating Tuples

Accessing Values in Dictionary

To access dictionary elements, you can use the familiar square brackets along with the key to obtain its value. Following is a simple example –

Live Demo

```
#!/usr/bin/python
```

```
dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}  
print "dict['Name']: ", dict['Name']  
print "dict['Age']: ", dict['Age']
```

When the above code is executed, it produces the following result –

```
dict['Name']: Zara  
dict['Age']: 7
```

If we attempt to access a data item with a key, which is not part of the dictionary, we get an error as follows –

Live Demo
#!/usr/bin/python

```
dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}  
print "dict['Alice']: ", dict['Alice']
```

When the above code is executed, it produces the following result –

```
dict['Alice']:  
Traceback (most recent call last):  
  File "test.py", line 4, in <module>  
    print "dict['Alice']: ", dict['Alice'];  
KeyError: 'Alice'
```

Updating Dictionary

You can update a dictionary by adding a new entry or a key-value pair, modifying an existing entry, or deleting an existing entry as shown below in the simple example –

Live Demo
#!/usr/bin/python

```
dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}  
dict['Age'] = 8; # update existing entry  
dict['School'] = "DPS School"; # Add new entry
```

```
print "dict['Age']: ", dict['Age']  
print "dict['School']: ", dict['School']
```

When the above code is executed, it produces the following result –

```
dict['Age']: 8  
dict['School']: DPS School  
Delete Dictionary Elements
```

You can either remove individual dictionary elements or clear the entire contents of a dictionary. You can also delete entire dictionary in a single operation.

To explicitly remove an entire dictionary, just use the del statement. Following is a simple example –

Live Demo

```
dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}  
del dict['Name']; # remove entry with key 'Name'  
dict.clear();    # remove all entries in dict  
del dict ;       # delete entire dictionary
```

```
print "dict['Age']: ", dict['Age']  
print "dict['School']: ", dict['School']
```

This produces the following result. Note that an exception is raised because after `del dict` dictionary does not exist any more –

```
dict['Age']:  
Traceback (most recent call last):  
  File "test.py", line 8, in <module>  
    print "dict['Age']: ", dict['Age'];  
TypeError: 'type' object is unsubscriptable  
Note – del() method is discussed in subsequent section.
```

Properties of Dictionary Keys

Dictionary values have no restrictions. They can be any arbitrary Python object, either standard objects or user-defined objects. However, same is not true for the keys.

There are two important points to remember about dictionary keys –

(a) More than one entry per key not allowed. Which means no duplicate key is allowed. When duplicate keys encountered during assignment, the last assignment wins.

7 Coding

python

```
from flask import Flask,render_template,request,redirect  
from flask_cors import CORS,cross_origin  
import pickle  
import pandas as pd  
import numpy as np  
  
app=Flask(__name__)  
cors=CORS(app)  
model=pickle.load(open('LinearRegressionModel.pkl','rb'))  
car=pd.read_csv('Cleaned_Car_data.csv')  
  
@app.route('/',methods=['GET','POST'])  
def index():  
    companies=sorted(car['company'].unique())  
    car_models=sorted(car['name'].unique())  
    year=sorted(car['year'].unique(),reverse=True)  
    fuel_type=car['fuel_type'].unique()
```

```

__companies.insert(0,'Select Company')

__return __render_template('index.html',companies=companies, __car_models=car_models,
years=year,fuel_types=fuel_type)

@app.route('/predict',methods=['POST'])

@cross_origin()

def predict():

__company=request.form.get('company')

__car_model=request.form.get('car_models')

__year=request.form.get('year')

__fuel_type=request.form.get('fuel_type')

__driven=request.form.get('kilo_driven')

__prediction=model.predict(pd.DataFrame(columns=['name', 'company', 'year', 'kms_driven',
'fuel_type'],

____data=np.array([car_model,company,year,driven,fuel_type]).reshape(1, 5)))

__print(prediction)

```

```
return str(np.round(prediction[0],2))
```

```
if __name__ == '__main__':
```

```
app.run()
```

Car Price Predictor

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<module type="PYTHON_MODULE" version="4">
```

```
<component name="NewModuleRootManager">
```

```
<content url="file://$MODULE_DIR$">
```

```
<excludeFolder url="file://$MODULE_DIR$/venv" />
```

```
</content>
```

```
<orderEntry type="inheritedJdk" />
```

```
<orderEntry type="sourceFolder" forTests="false" />
```

```
</component>
```

```
</module>
```

misc.xml

<?xml version="1.0" encoding="UTF-8"?>

<project version="4">

<component name="ProjectRootManager" version="2" project-jdk-name="Python 3.8 (Car Price Predictor)" project-jdk-type="Python SDK" />

<component name="PyCharmProfessionalAdvertiser">

<option name="shown" value="true" />

</component>

</project>

vcs.xml

<?xml version="1.0" encoding="UTF-8"?>

<project version="4">

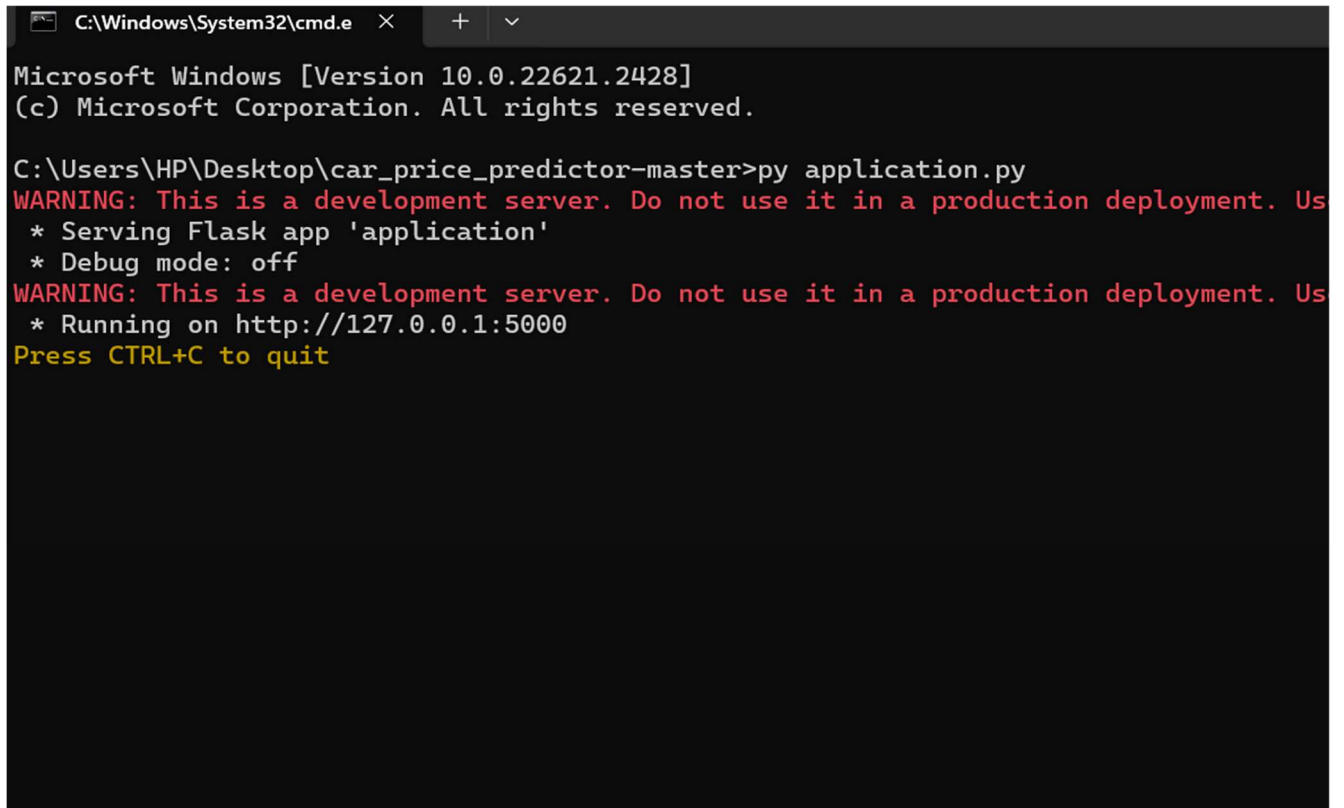
<component name="VcsDirectoryMappings">

<mapping directory="\$PROJECT_DIR\$" vcs="Git" />

</component>

</project>

8.OUTPUT SCREENS



```
C:\Windows\System32\cmd.e  X  +  v

Microsoft Windows [Version 10.0.22621.2428]
(c) Microsoft Corporation. All rights reserved.

C:\Users\HP\Desktop\car_price_predictor-master>py application.py
WARNING: This is a development server. Do not use it in a production deployment. Us
* Serving Flask app 'application'
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Us
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
```


the app predicts the price of a car you want to buy by filling the details below.

Select the company:

Audi

Select Company

Audi

BMW

Chevrolet

Datsun

Fiat

Force

Ford

Hindustan

Honda

Hyundai

Jaguar

Jeep

Land

Mahindra

Maruti

Mercedes

Mini

Mitsubishi

Nissan

Select the company:

Audi

Select the model:

Audi A3 Cabriolet

Audi A3 Cabriolet

Audi A4 1.8

Audi A4 2.0

Audi A6 2.0

Audi A8

Audi Q3 2.0

Audi Q5 2.0

Audi Q7

Enter the Number of Kilometres that the car has travelled:

Enter the kilometres driven

Predict Price

MINI-PRO

2019

2018

2017

2016

2015

2014

2013

2012

2011

2010

2009

2008

2007

2006

2005

2004

2003

2002

2001

2019

Select the Fuel Type:

Petrol

Enter the Number of Kilometres that the car has travelled:

Enter the kilometres driven

Select the company:

Audi

Select the model:

Audi A3 Cabriolet

Select Year of Purchase:

2019

Select the Fuel Type:

LPG

Petrol

Diesel

LPG

Enter the kilometres driven

Predict Price

Select the company:

Audi

Select the model:

Audi A3 Cabriolet

Select Year of Purchase:

2019

Select the Fuel Type:

LPG

Enter the Number of Kilometres that the car has trav

20000

Audi

Select the model:

Audi A3 Cabriolet

Select Year of Purchase:

2019

Select the Fuel Type:

LPG

Enter the Number of Kilometres that the car has travelled:

20000

Predict Price

Prediction: ₹3205696.45

Select the company:

Honda

Select the model:

Honda Accord

Select Year of Purchase:

2016

Select the Fuel Type:

LPG

Enter the Number of Kilometres that the car has travelled:

098788

Predict Price

Prediction: ₹330158.71

This app predicts the price of a car you want to sell. Try filling the details below:

Select the company:

Select Company

Select the model:

Select Year of Purchase:

2019

Select the Fuel Type:

Petrol

Enter the Number of Kilometres that the car has travelled:

Enter the kilometres driven

Predict Price

9.TESTING

SYSTEM TEST

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

TYPES OF TESTS

Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application. It is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfactory, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

Functional test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

- Valid Input : identified classes of valid input must be accepted.
- Invalid Input : identified classes of invalid input must be rejected.
- Functions : identified functions must be exercised.
- Output : identified classes of application outputs must be exercised.
- Systems/Procedures : interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

Unit Testing

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

9.1 Sample Test Cases:

Step	Test Case	Test Data	Expected Result	Actual Result	Status (Pass/Fail)
1	Launch Application	http://127.0.0.1:5000/	Main page of Application	Main page of Application	Pass

Step	Test Case	Test Data	Expected Result	Actual Result	Status (Pass/Fail)
3	Prediction of Used Car Prices	http://127.0.0.1:5000/ predicts/	Prediction of Used Car Prices	Prediction of Used Car Prices	Pass

10 CONCLUSION:

The increased prices of new cars and the financial incapability of the customers to buy them, Used Car sales are on a global increase. Therefore, there is an urgent need for a Used Car Price Prediction system which effectively determines the worthiness of the car using a variety of features. The proposed system will help to determine the accurate price of used car price prediction. This paper compares 3 different algorithms for machine learning : Random forest Regression, Lasso Regression and Ridge Regression.

11.Future Enhancement

Incorporate More Data Sources: Expanding the data sources beyond a single online seller, such as "Njuškalo," can provide a more comprehensive view of the used car market. Integration with data from multiple sources, including dealerships, auctions, and private sellers, can enrich the dataset and improve model accuracy.

Feature Engineering: Investigate and develop more sophisticated feature engineering techniques. Additional features, such as vehicle history, maintenance records, and accident reports, could be incorporated to further refine the predictive models.

Time Series Analysis: Consider time series analysis to capture temporal trends in used car prices. Analyzing how prices change over months, seasons, or years can provide deeper insights into market dynamics and help refine predictions.

Geospatial Analysis: Integrate geospatial data to examine the impact of location on used car prices. Different regions or cities may have varying price trends, and this information can be valuable for both buyers and sellers.

Advanced Machine Learning Techniques: Explore more advanced machine learning algorithms, including ensemble methods, deep learning, and natural language processing. These techniques may improve predictive accuracy and handle unstructured data such as vehicle descriptions and images.

Hybrid Models: Combine multiple machine learning models to create ensemble models that leverage the strengths of different algorithms. This can lead to more accurate predictions and better generalization.

Real-Time Pricing: Develop a real-time pricing model that continuously updates predictions based on new market data. This can assist sellers in determining the optimal time to list their vehicles and buyers in making purchase decisions.

Explainable AI: Implement explainable AI techniques to make the model's predictions more interpretable. This is especially important for building trust among users and understanding the factors driving price predictions.

Mobile pplications: Create user-friendly mobile applications or web tools that allow consumers to access price predictions, facilitating their decision-making process when buying or selling used cars.

Marketplace Integration: Collaborate with online marketplaces or used car listing platforms to integrate the predictive model, enabling users to obtain price estimates when listing their vehicles.

Data Privacy and Security: Address data privacy concerns by implementing robust data anonymization techniques, ensuring that sensitive information is protected when collecting and analyzing data from various sources.

Validation and Cross-Validation: Rigorously validate the models using cross-validation techniques and evaluate their performance under different market conditions to ensure robust and reliable predictions.

Research on Interpretability: Conduct research on interpretability methods for machine learning models to better understand how various features affect price predictions. This can help users make more informed decisions.

Customer Feedback and User Experience: Gather user feedback and continuously improve the models and applications based on user experiences and needs.

Regulatory Compliance: Stay updated with local and national regulations related to used car sales and pricing, ensuring that the developed models adhere to legal and ethical standards.

12.REFERENCES:

- [1]Sinha, S. a. Azim, R. a. Das and Sourav, "Linear Regression on Car Price Prediction," 2020.
- [2]Prediction of Used Car Prices Using Machine Learning Techniques" by M. I. A. Miah, M. S. Arefin, and M. S. Islam (2022).
- [3]Predicting Used Car Prices Using Neural Networks" by A. F. Rahman, M. A. Uddin, and M. M. Islam (2020).
- [4]Gegic, E. et al. (2019) demonstrate the need to create a model to forecast the cost of second hand cars in Bosnia and Herzegovina.
- [5] S. Pudaruth, “Predicting the Price of Used Cars using Machine Learning Techniques,” International Journal of Information & Computation Technology, vol. 4, no. 7, pp. 753–764, 2014.
- [6] N. Kanwal and J. Sadaqat, “Vehicle Price Prediction System using Machine Learning Techniques,” International Journal of Computer Applications, vol. 167, no. 9, pp. 27–31, 2017.
- [7] S. Peerun, N. H. Chummun, and S. Pudaruth, “Predicting the Price of Second-hand Cars using Artificial Neural Networks,” The Second International Conference on Data Mining, Internet Computing, and Big Data, no. August, pp. 17–21, 2015.
- [8] N.Sun, H. Bai, Y. Geng, and H. Shi, “Price evaluation model in second-hand car system based on BP neural network theory,” in 2017 18th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), jun 2017, pp. 431–436.

- [9] G.Rossum, “Python Reference Manual,” Amsterdam, The Netherlands, The Netherlands, Tech. Rep., 1995.
- [10] A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios, “Duplicate Record Detection: A Survey,” IEEE Transactions on Knowledge and Data Engineering, vol. 19, no. 1, pp. 1–16, jan 2007.
- [11] G.Chandrashekar and F. Sahin, “A survey on featureselection methods,” Computers & Electrical Engineering, vol. 40, no. 1, pp. 16–28, 2014. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0045790613003066>
- [12]M.C.Newman,“Regression analysis of log-transformed data: Statistical bias and its correction,” Environmental Toxicology and Chemistry, vol. 12, no. 6, pp. 1129–1133, 1993. [Online]. Available: <http://dx.doi.org/10.1002/etc.5620120618>
- [13] R.Taylor, “Interpretation of the Correlation Coefficient: A Basic Review,” Journal of Diagnostic Medical Sonography, vol. 6, no. 1, pp. 35–39, 1990.
- [14] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vander- (5)plas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duches-nay, “Scikit-learn: Machine Learning in fPgython,” Journal of Machine Learning Research, vol. 12, pp. 2825–2830, 2011.
- [15] J. Morgan, “Classification and Regression Tree Analy-sis,” Bu.Edu, no. 1, p. 16, 2014. [Online]. Available: <http://www.bu.edu/sph/files/2014/05/MorganCART.pdf>